La bibliothèque d'algébre linéaire récente de Maple est `LinearAlgebra`. (Une bibiothèque plus ancienne `linalg` est également présente). L'appel se fait avec

```
with(LinearAlgebra):
```

Il peut être utile d'appeler cette bibliothèque avec un ";" pour afficher la liste des fonctions. On ne donne ici qu'une introduction très limitée à cette bibliothèque extrèmement riche. La plupart des fonctions de `LinearAlgebra` commencent par une majuscule (souvent il y a d'autres majuscules au milieu du nom) contrairement à celles de `linalg`.

1. Algèbre linéaire générale

Constructeurs Plusieurs formes possibles.

- Pour des matrices lignes ou colonnes
  ```
  > <a|b|c>;<a,b,c>;
  ```
  Les matrices colonnes sont aussi désignées par le terme de « vecteur »dans l'aide de Maple.
- Pour une matrice d'un autre type
  ```
  > A:=RandomMatrix(3);
    B:=<<a|b|c>,<u|v|w>>;
    M := <<1,1,1,4>|<1,1,-2,1>|<3,1,1,8>>;
    C:= Matrix(2,3,[[1,2,3],[l,j,g]]);
  ```
  Le constructeur `Matrix` offre une myriade de possibilités. Consulter l'aide. Attention, les types de données de cette bibliothèque ne coïncident pas tout à fait aux types d'objets mathématiques. En particulier une colonne (constructeur `< , , >`) *n'est pas* une matrice à une colonne (constructeur `Matrix(p,1)`). De même, une ligne (constructeur `< >` avec des | comme séparateurs) *n'est pas* une matrice à une ligne (constructeur `Matrix(1,q)`).

Addition Le + habituel convient

```
> A:=RandomMatrix(3);
  B:=RandomMatrix(3);
  A+B;
```

Multiplication externe Le * habituel convient.

```
> A:=RandomMatrix(3); lambda*A;
```

Multiplication matricielle Un opérateur "magique" : le point "." (dot). Il fait à peu près ce qu'on voudrait qu'il fasse. On peut l'employer comme le "rien" que l'on place entre deux matrices lorsqu'un produit est possible.

```
> A:=RandomMatrix(3);
  C:=<a,b,c>; L:=<u|v|w>;
  A.C; L.A; L.C; C.L;
```

Extraire des données Pour un terme `A[1,2]`. Pour extraire une matrice

```
> A:=RandomMatrix(5);
  SubMatrix(A, [2..5], [2..3, 1]);
```

Déterminant Simplement `Determinant(M)` avec majuscule et sans accent.

Trace Simplement `Trace(M)` avec majuscule.

Rang Simplement `Rank(M)`.

Système linéaire La syntaxe présentée ici est basique, la procédure renvoie les solutions (avec d'éventuels paramètres) d'une équation matricielle $AX = b$ d'inconnue une matrice colonne X avec un second membre b qui est une matrice colonne.

```
> A := <<1,0,0>|<2,1,0>|<1,0,0>|<-1,-1,-3>>;
  b := <2,-1,-9>:
  LinearSolve(A, b);
  b := Matrix(3,1);
  LinearSolve(A, b);
```

Consultez l'aide de `LinearSolve()`, les possibilités sont innombrables.

2. Algèbre linéaire euclidienne

Produit scalaire La fonction `DotProduct` calcule le produit scalaire canonique de deux matrices colonnes de n'importe quelle taille.

```
> U:=<a,b,c>; V:=<u,v,w>;
  DotProduct(U,V);
```

Comme toujours dans Maple, par défaut les variables sont à valeurs complexes et `DotProduct` calcule les produit hermitien en conjuguant automatiquement la colonne de gauche.
On peut ajouter un paramètre matriciel `A` à l'appel de `Bilinear` (voir aide) cela permet de calculer un produit scalaire en prenant les coordonnées dans une base qui n'est pas orthonormée.

Produit vectoriel La fonction `CrossProduct` calcule le produit vectoriel de deux matrices colonnes de 3 lignes.

```
> U:=<a,b,c>; V:=<u,v,w>;
  CrossProduct(U,V);
```